FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Interner Bericht

AIX Tipps und Tricks

Klaus Wolkersdorfer

FZJ-ZAM-IB-2002-04

März 2002

(letzte Änderung: 26.03.2002)

Vortrag: AIX-Arbeitskreis, 28.3.2002, Heidelberg

AIX Tipps und Tricks

Klaus Wolkersdorfer

(K.Wolkersdorfer@fz-juelich.de)

Forschungszentrum Jülich GmbH

Zentralinstitut für Angewandte Mathematik (ZAM)

Tel: +49-2461-61-6579

Content

- Resource Control
  - Virtual storage (32 bit + 64 bit)
  - CPU
- Compiler
- Tuning
- Performance Tools

Resource Control: Virtual Storage

Limits in 32 bit mode (Compiler-Default-Option: -q32):

- stack: 256MB (automatic storage, -qnosave)
  only 1 segment register: 0x2
- data: 2GB (static storage, -qsave, -bmaxdata:…)
  8 segment register: 0x3-0xA
  To patch a binary to use all 8 segment registers:
  /usr/bin/echo '\0200\0\0\0' | dd of=… bs=4 count=1 seek=19 conv=notrunc

Resource Control: Virt.Storage (cont.)

Limits in 64 bit mode (-q64):

- stack: 64PB (segments: 0xF000 0000 – 0xFFFF FFFF)
- data: 384PB (segments: 0x0000 0010 – 0x6FFF FFFF)
- No need for –bmaxdata/-bmaxstack (without soft limits)
- Recommended options: -qwarn64, -qintsize=8
- See: *AIX 64-bit Performance in Focus (SG24-5103)*

Resource Control: Virt.Storage (cont.)

- Important for machines with large memory
- System crash after user started a job with 18GB on our machine with 8GB memory and 16GB paging space
- First Idea: Use data_hard/stack_hard in /etc/security/limits
  ➔ More problems

Resource Control: Virt.Storage (cont.)

- If data_hard < 700MB
  ➔ Compiler crash with 0509-036/026
- If 700MB < data_hard < 2GB
  ➔ Compiler warning message: 1501-245,1540-2001(C++)
  (suppress it via /etc/xlf.cfg: -qsuppress=1501-245)
  For C++ see APAR IY25384
- If 2GB < data_hard
  ➔ AIX treats it as unlimited (even AIX 5.1)

Resource Control: Virt.Storage (cont.)

- Conclusion: **Virtual storage limitation impossible in AIX**
- WLM useful for resource management but not for limitation
- From PMR 47366,033,724: e-fix for libc.a was provided
  "… you are the only one to have asked for higher limits…"
  "…design change request (**DCR: MR031302356**) opened …"
- Major defect for a modern 64 bit operating system

Resource Control: CPU

- In /etc/security/limits: cpu_hard
- Works only for ***user*** cpu time (not ***system*** time)
  Example: Did you ever catch a looping netscape process?
- Can only control total CPU for all threads of a process
- What about control number of thread creation?

Resource Control: CPU (cont.)

- User can create thousands of threads i.e. via
  - export XLSMPOPTS=parthds=…
  - in Fortran/C: omp_set_num_threads(…)
- Such a user **will** monopolize any SMP machine (Again: WLM cannot help here)
- This is fair in a single user environment **only**
- Multi user SMP machines have a problem

OpenMP FORTRAN Example

- Compile with: xlf_r –qsmp –q64 …

```
!$OMP PARALLEL DO
        do K=1,omp_get_max_threads()
        …
        end do
 !$OMP END PARALLEL DO
```

OpenMP FORTRAN Example (cont.)

- Each thread needs stack space
- Controlled via
  export XLSMPOPTS=stack=…
- XLF Users Guide: Default is 4 194 304
- **Doc-Error** in C Users Guide: **32768 is wrong**!

Compiler: LUM

- Problems (segmentation fault) with AIX 5.1 and LUM:
  solved with ifor_ls…5.1.0.16 and last C/C++ upgrade
- Recommendation: **Do not use it!**
- Instead use:
  - Config-file: -qnolm (for batch compiler)
  - Env. variable: export NOLUM=1 (for vacide)

Compiler: Local Configuration

- Done via NIM bundles
- Additional options in /etc/xlf.cfg:
  -qnolm,-qhalt=E,-qmaxmem=16384,-L/usr/local/lib
- Additional options in /etc/vac.cfg:
  -qnolm,-qmaxerr=8:s,-L/usr/local/lib
- Recently added to both: -qsuppress=1501-245

Compiler: Remote Doc. Server

- See: *AIX 5L Porting Guide* (SG24-6034) 4.11.2
- It works !!! **But…**
- Segmentation fault with AIX 5.1 (IMNSearch…2.1.3.0):
- PMR 46383,033,724: No solution yet
- different search results in AIX 4.3.3 and AIX 5.1
- watch for /var/docsearch/cgi-bin/core after every search

Tuning: Paging

- Deferred Paging Space Allocation (versus: late and early)
- Small /dev/hd6 on mirrored rootvg
- Larger area(s) on not mirrored, not busy disks
- Starting with AIX 5.1: After boot is complete:
  swapoff /dev/hd6
- To relax busy rootvg

Tuning: Large Memory Tuning

- Motivation: File buffers filling up all memory
  ➔ No space left for programs
  ➔ System steals pages (bad performance)
- Default is good for I/O, bad for large programs (Oracle!)
- vmtune –p 5 –P 10 –h 1
  makes 5% a low limit and 10% a high limit for file buffers
- -h 1 insures the high limit is a hard limit

Tuning: Large Memory Tuning (cont.)

- vmtune –W 64
- Threshold for random writes to accumulate in RAM before subsequent pages are flushed to disk
- Limits dirty pages in memory
- Reduces system overhead
- Favors interactive response time over throughput

Tuning: no and nfso commands

- Changes must be within /etc/rc.net
- After AIX 4.3.3: use default for *thewall* and *sb_max*
- no –o udp_sendspace=65536 -o udp_recvspace=262144 depends on adapter used (send: >65536 is ineffective)
- no –o tcp_pmtu_discover=0 -o udp_pmtu_discover=0 prohibits permanent changes of routing table in JuNet
- nfso –o nfs_socketsize=262144

Tuning: Disk I/O Pacing

- Problem: One I/O task, i.e. *cp* can take several minutes without letting somebody (i.e. compile, vi) in between
- Very important for multiuser systems
- After several tests we found: **Rule of 17 and 4:**
  chsys –l sys0 –a maxpout=17 –a minpout=4
- Could also be done via *smitty chgsys*
- Prohibits one large I/O to monopolize system

Performance Tools

- *AIX 5L Performance Tools Handbook (SG24-6039)*
- Perfstat API: libperfstat.a (fileset: bos.perf.perfstat)
  /usr/samples/libperfstat/perfsample.c
- Performance Monitor API: libpmapi.a (bos.pmapi.lib)
- System Performance Measurement Interface (SPMI)
  libSpmi.a (perfagant.tools)

Segal's Law:

'a man with one watch knows what time it is

…a man with two watches is never sure'